Model Building in Polaris: A Deeper Dive

Updated October 2025

Anaplan

Anaplan Polaris is our next-generation calculation engine.

Designed from the ground up for fast, highly dimensioned calculation at scale, Polaris represents a paradigm shift in your ability to model business planning challenges without compromise.

You can achieve significantly greater granularity of planning for more detailed decision-making with Polaris.

This deck is part of a series on Polaris best practices and includes a deep dive of some previously discussed topics, as well as new content.

Click here for more Community content or visit Anapedia for detailed technical guidance.

Topics Covered

1 Iterative Development

2 Blueprint Insights: Better Together

3 Calculation Complexity

4 Calculation Effort

5 Guards and Inline Conditions

6 Inserting List Members

Iterative Development

An introduction to Iterative Development in Polaris

Polaris allows large-scale modeling, requiring phased validation to stay performant and manageable

What is iterative development?

Iterative development is **separating syntax validation from performance validation.**

It means building and unit testing in one environment, evaluating scalability and performance in another, and iterating between the two throughout.

Why does it matter?

This development approach helps you **build correct and performant models**, **while maintaining model builder productivity**.

Following this approach using Anaplan's Application Lifecycle Management (ALM) **establishes deployment discipline** from the start.

What does it look like?

There are four main steps:

1. Quick feedback, fast development
2. Create a TEST model with fully populated lists (don't forget to put it in deployed mode!)
3. Start introducing data
4. Reality check with a full-scale test model

Read this 5-minute article for a closer look at iterative development.

Blueprint Insights: Better Together

Getting the most out of Blueprint Insights

Blueprint Insights can and should be used together to provide a holistic view into optimization opportunities

Calculation Complexity

Shows the effect of formulas in calculating cells

Calculation Effort

Displays the percentage of computational effort required for each line item

Populated Cell Count

Tells the size of the populated space - the cells that have non-default values

Cell Count

Shows the total size of the multi-dimensional space (the potentially addressable space)

Prioritizing Optimization Opportunities

Review the below guidelines for prioritizing which formulas to optimize in a Polaris model. Then explore the remaining content for more detail about Calculation Complexity, Calculation Effort, and Using Guards Effectively to Optimize Calculations.

1. Focus first on **high Calculation Effort** line items that also have **high Calculation Complexity**. These line items have the most room for improvement.

2. Focus next on line items with **high Calculation Effort overall**, especially if they use known single-threaded functions like ISFIRSTOCCURRENCE, RANK, RANKCUMULATE, and CUMULATE.

3. Focus next on **high Calculation Complexity line items that also have high Cell Count**. Even if they have a small Populated Cell Count and Calculation Effort, they have the potential to cause performance issues if the addressable Cell Count and Complexity is high.

4. Finally, focus on line items with **Calculation Complexities that equal 'All Cells' in line items with high cell counts** as they may pose a risk due to their inherent fully dense nature.

Calculation Complexity

The Calculation Complexity column shows the effect of your formula in calculating cells

Formulas should be designed to preserve sparsity to drive memory-efficient models

One-to-One

Only calculates results for cells that contain a non-default value

Therefore, the number of cells calculated in the result is the same as non-default values in the source

One-to-Many

Calculates results for a greater number of cells than make up the source data

The multiplier, or ratio, is a measure of how much work the engine must do relative to the number of populated cells in the source – the higher this is, the less efficient the formula will be

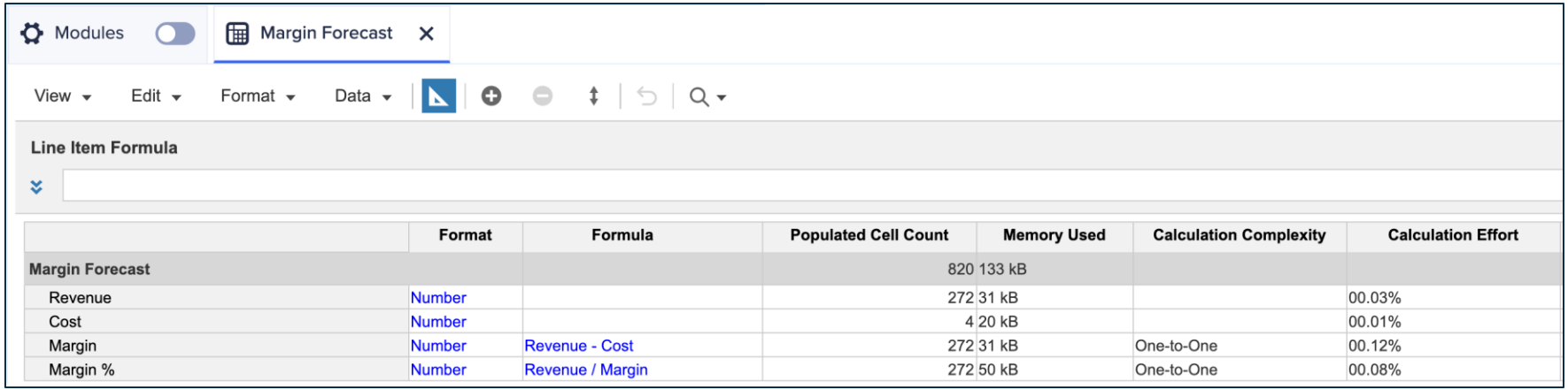
All cells

Calculates results for all cells within the dimensionality of the line item

Every cell in the result requires calculation, even those that were originally blank or evaluated to be zero

‘One-to-One’ Calculation Complexity: An example

In this example, the Margin calculation is driven by the Revenue value and influenced by the Cost value, representing a one-to-one relationship. The one-to-one calculation complexity means one cell is calculated and potentially populated for each non-default value in the source.



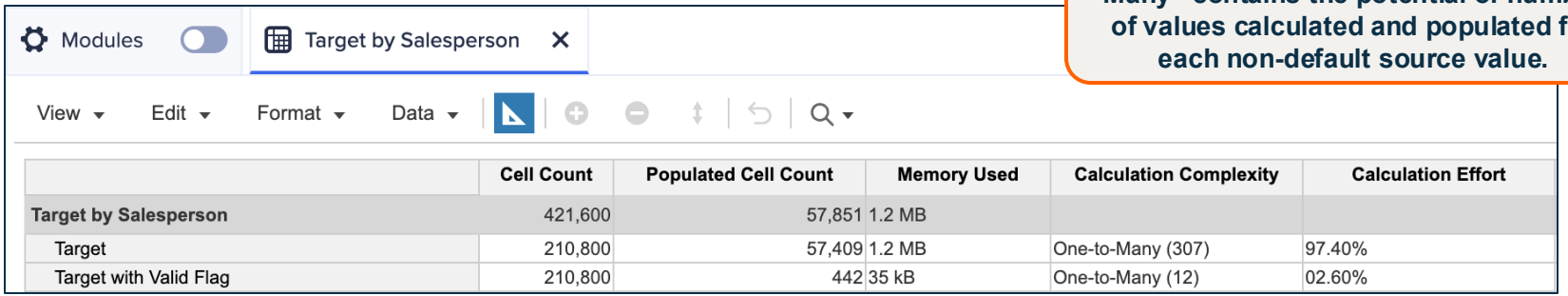
	Format	Formula	Populated Cell Count	Memory Used	Calculation Complexity	Calculation Effort
Margin Forecast			820	133 kB		
Revenue	Number		272	31 kB		00.03%
Cost	Number		4	20 kB		00.01%
Margin	Number	Revenue - Cost	272	31 kB	One-to-One	00.12%
Margin %	Number	Revenue / Margin	272	50 kB	One-to-One	00.08%

One-to-one calculations are common, but there are other types of math, such as allocations, that do not follow this pattern.

‘One-to-Many’ Calculation Complexity: An example

The One-to-Many calculation arises when math is performed that differs dimensionally from the source data or involves allocations. The below example looks at the target by salesperson and explores two ways to write the formula.

Note: The parenthetical after “One-to-Many” contains the potential of number of values calculated and populated for each non-default source value.



	Cell Count	Populated Cell Count	Memory Used	Calculation Complexity	Calculation Effort
Target by Salesperson	421,600	57,851	1.2 MB		
Target	210,800	57,409	1.2 MB	One-to-Many (307)	97.40%
Target with Valid Flag	210,800	442	35 kB	One-to-Many (12)	02.60%

Each line item results in the target by salesperson, but the formula is written two different ways.

CONSIDER: What do you notice about the cell counts, memory, complexity, and effort for each line item? Why might this be?

‘One-to-Many’ Calculation Complexity: An example (continued)

The initial (top) formula results in 307 potential values for every non-default value feeding it. By introducing a guard to the formula, in this case the ‘IF’ statement, the second (bottom) formula results in a complexity of 12, optimizing formula efficiency and reducing unnecessary calculations.

Target by Salesperson — Target POLARIS

```
Margin Forecast.Revenue * (1 + Overassign %.to Salesperson) / Salespeople Count per Product.Count
```

Cell Count: 210,800
Populated Cell Count: 57,409
Memory Used: 1.2 MB
Calculation Complexity: One-to-Many (307)
Calculation Effort: 97.4%

Target by Salesperson — Target with Valid Flag POLARIS

```
IF
    Products by Salesperson.Valid Product
THEN
    Margin Forecast.Revenue * (1 + Overassign %.to Salesperson) / Salespeople Count per Product.Count
ELSE
    0
```

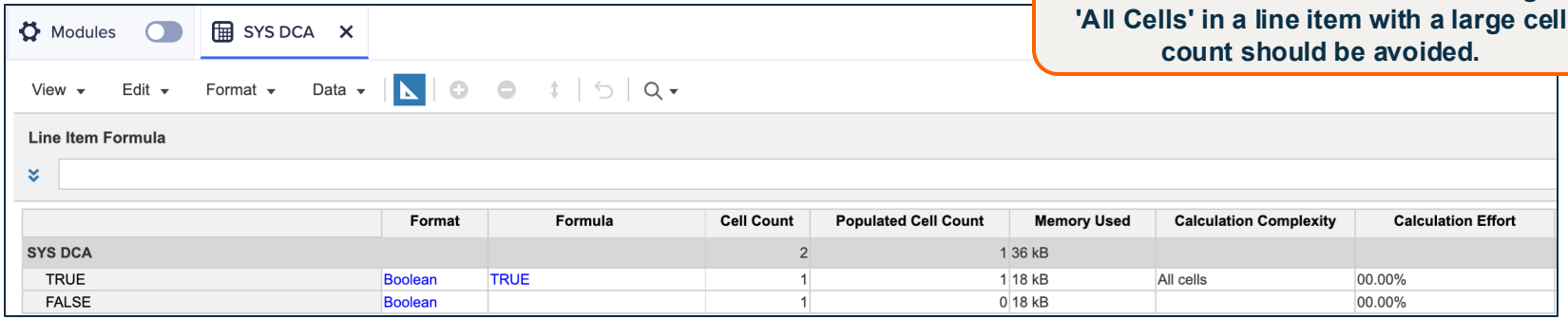
Cell Count: 210,800
Populated Cell Count: 442
Memory Used: 35 kB
Calculation Complexity: One-to-Many (12)
Calculation Effort: 02.60%

BEST PRACTICE: To optimize formula efficiency and reduce unnecessary calculations, minimize the use of One-to-Many calculations by reviewing the differing dimensionality, and/or using guards to lower the complexity factor (see "Guards" section for more detail).

‘All cells’ Calculation Complexity: An example

In this example, a Boolean is used where the formula consists of the value TRUE, which applies to all cells. When dealing with a substantial number of cells, the ‘All cells’ approach can negatively impact performance as the engine must calculate the value for each cell individually.

Note: Cell Count is the important insight to focus on here. A formula resulting in 'All Cells' in a line item with a large cell count should be avoided.



	Format	Formula	Cell Count	Populated Cell Count	Memory Used	Calculation Complexity	Calculation Effort
SYS DCA			2	1	36 kB		
TRUE	Boolean	TRUE	1	1	18 kB	All cells	00.00%
FALSE	Boolean		1	0	18 kB		00.00%

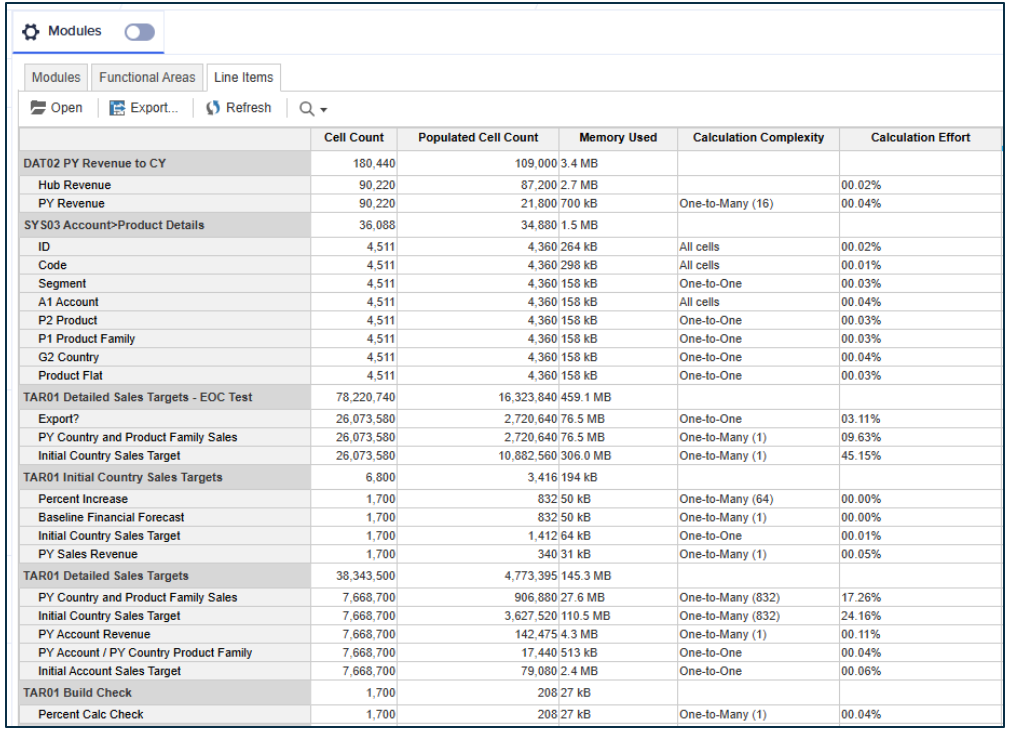
BEST PRACTICE: Avoid the ‘All cells’ approach in scenarios involving a substantial number of cells where performance is critical.

Watch this 5-minute video for a deep dive on **Calculation Complexity**.

Calculation Effort

How to determine which line items to optimize?

Blueprint Insights contain valuable information about line items, such as Calculation Effort. When reviewed together, Blueprint Insights can provide valuable information so you can focus on optimizing the line items that will have the greatest impact on performance.



Modules

Modules | Functional Areas | Line Items

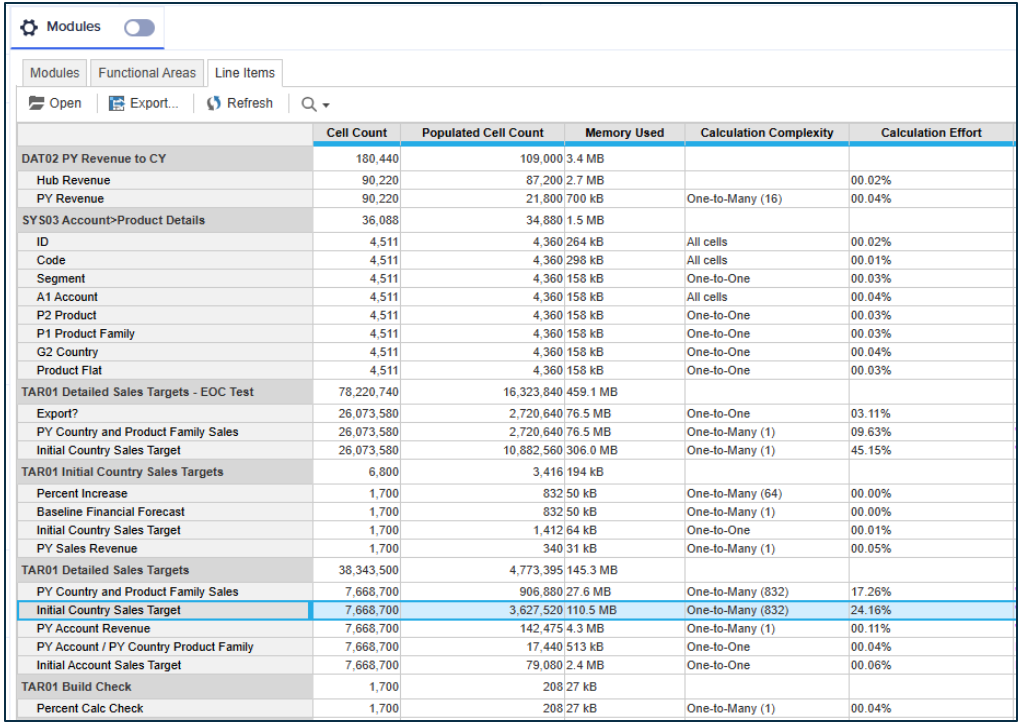
Open | Export... | Refresh

	Cell Count	Populated Cell Count	Memory Used	Calculation Complexity	Calculation Effort
DAT02 PY Revenue to CY	180,440	109,000	3.4 MB		
Hub Revenue	90,220	87,200	2.7 MB		00.02%
PY Revenue	90,220	21,800	700 kB	One-to-Many (16)	00.04%
SYS03 Account>Product Details	36,088	34,880	1.5 MB		
ID	4,511	4,360	264 kB	All cells	00.02%
Code	4,511	4,360	298 kB	All cells	00.01%
Segment	4,511	4,360	158 kB	One-to-One	00.03%
A1 Account	4,511	4,360	158 kB	All cells	00.04%
P2 Product	4,511	4,360	158 kB	One-to-One	00.03%
P1 Product Family	4,511	4,360	158 kB	One-to-One	00.03%
G2 Country	4,511	4,360	158 kB	One-to-One	00.04%
Product Flat	4,511	4,360	158 kB	One-to-One	00.03%
TAR01 Detailed Sales Targets - EOC Test	78,220,740	16,323,840	459.1 MB		
Export?	26,073,580	2,720,640	76.5 MB	One-to-One	03.11%
PY Country and Product Family Sales	26,073,580	2,720,640	76.5 MB	One-to-Many (1)	09.63%
Initial Country Sales Target	26,073,580	10,882,560	306.0 MB	One-to-Many (1)	45.15%
TAR01 Initial Country Sales Targets	6,800	3,416	194 kB		
Percent Increase	1,700	832	50 kB	One-to-Many (64)	00.00%
Baseline Financial Forecast	1,700	832	50 kB	One-to-Many (1)	00.00%
Initial Country Sales Target	1,700	1,412	64 kB	One-to-One	00.01%
PY Sales Revenue	1,700	340	31 kB	One-to-Many (1)	00.05%
TAR01 Detailed Sales Targets	38,343,500	4,773,395	145.3 MB		
PY Country and Product Family Sales	7,668,700	906,880	27.6 MB	One-to-Many (832)	17.26%
Initial Country Sales Target	7,668,700	3,627,520	110.5 MB	One-to-Many (832)	24.16%
PY Account Revenue	7,668,700	142,475	4.3 MB	One-to-Many (1)	00.11%
PY Account / PY Country Product Family	7,668,700	17,440	513 kB	One-to-One	00.04%
Initial Account Sales Target	7,668,700	79,080	2.4 MB	One-to-One	00.06%
TAR01 Build Check	1,700	208	27 kB		
Percent Calc Check	1,700	208	27 kB	One-to-Many (1)	00.04%

CONSIDER: Of the line items shown, which Blueprint Insights stand out? Which line items might have the greatest opportunity for optimization?

Use Calculation Effort with Calculation Complexity to optimize formulas

Calculation Effort and Calculation Complexity are Blueprint Insights contained within the blueprint view of a module (or the line items tab) in Polaris. Calculation effort displays the percentage of computational effort for each line item over the last ten minutes and responds to formula changes or user inputs.



	Cell Count	Populated Cell Count	Memory Used	Calculation Complexity	Calculation Effort
DAT02 PY Revenue to CY	180,440	109,000	3.4 MB		
Hub Revenue	90,220	87,200	2.7 MB		00.02%
PY Revenue	90,220	21,800	700 kB	One-to-Many (16)	00.04%
SYS03 Account>Product Details	36,088	34,880	1.5 MB		
ID	4,511	4,360	264 kB	All cells	00.02%
Code	4,511	4,360	298 kB	All cells	00.01%
Segment	4,511	4,360	158 kB	One-to-One	00.03%
A1 Account	4,511	4,360	158 kB	All cells	00.04%
P2 Product	4,511	4,360	158 kB	One-to-One	00.03%
P1 Product Family	4,511	4,360	158 kB	One-to-One	00.03%
G2 Country	4,511	4,360	158 kB	One-to-One	00.04%
Product Flat	4,511	4,360	158 kB	One-to-One	00.03%
TAR01 Detailed Sales Targets - EOC Test	78,220,740	16,323,840	459.1 MB		
Export?	26,073,580	2,720,640	76.5 MB	One-to-One	03.11%
PY Country and Product Family Sales	26,073,580	2,720,640	76.5 MB	One-to-Many (1)	09.63%
Initial Country Sales Target	26,073,580	10,882,560	306.0 MB	One-to-Many (1)	45.15%
TAR01 Initial Country Sales Targets	6,800	3,416	194 kB		
Percent Increase	1,700	832	50 kB	One-to-Many (64)	00.00%
Baseline Financial Forecast	1,700	832	50 kB	One-to-Many (1)	00.00%
Initial Country Sales Target	1,700	1,412	64 kB	One-to-One	00.01%
PY Sales Revenue	1,700	340	31 kB	One-to-Many (1)	00.05%
TAR01 Detailed Sales Targets	38,343,500	4,773,395	145.3 MB		
PY Country and Product Family Sales	7,668,700	906,880	27.6 MB	One-to-Many (832)	17.26%
Initial Country Sales Target	7,668,700	3,627,520	110.5 MB	One-to-Many (832)	24.16%
PY Account Revenue	7,668,700	142,475	4.3 MB	One-to-Many (1)	00.11%
PY Account / PY Country Product Family	7,668,700	17,440	513 kB	One-to-One	00.04%
Initial Account Sales Target	7,668,700	79,080	2.4 MB	One-to-One	00.06%
TAR01 Build Check	1,700	208	27 kB		
Percent Calc Check	1,700	208	27 kB	One-to-Many (1)	00.04%

HOW TO REFRESH CALCULATION EFFORT:

1. **Full Model Refresh:** close and reopen the model.
2. **Targeted Update:** allow model to remain idle for 10 minutes, then run the import actions or perform the user inputs. The Calculation Effort column on the line items affected will be updated.

UNSURE HOW TO PRIORITIZE WHICH LINE ITEMS TO OPTIMIZE?

Check out this slide for guidance on how to prioritize based on Blueprint Insights.

Read this 5-minute article for more information on **Calculation Effort**.

Using Guards Effectively

What is a “guard”?

A “guard” refers to an IF THEN statement used in a formula with conditions to tell the Polaris engine when to calculate. When used appropriately, guards will tell the engine when to calculate and when to skip, which contributes to a more performant model.

Community Test Module — Formula with Guards POLARIS

```
IF
    '<SOMETHING>'
THEN
    '<RESULT>'
ELSE
    '<DEFAULT VALUE>'
```

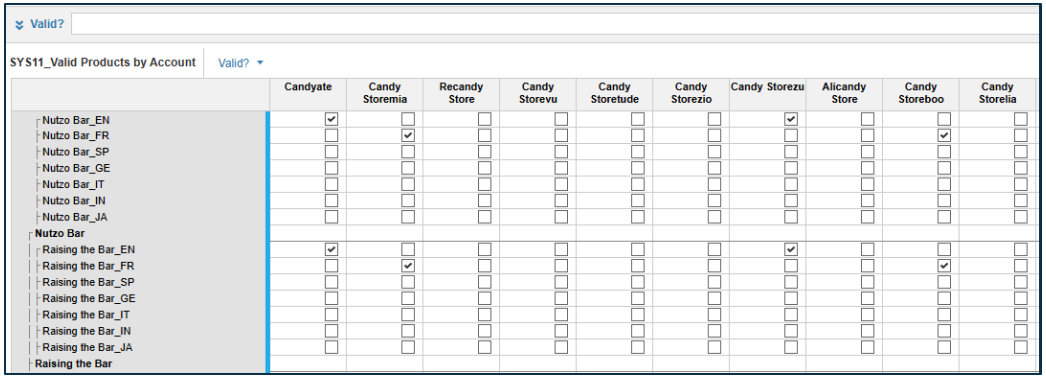
POLARIS IGNORES DEFAULTS

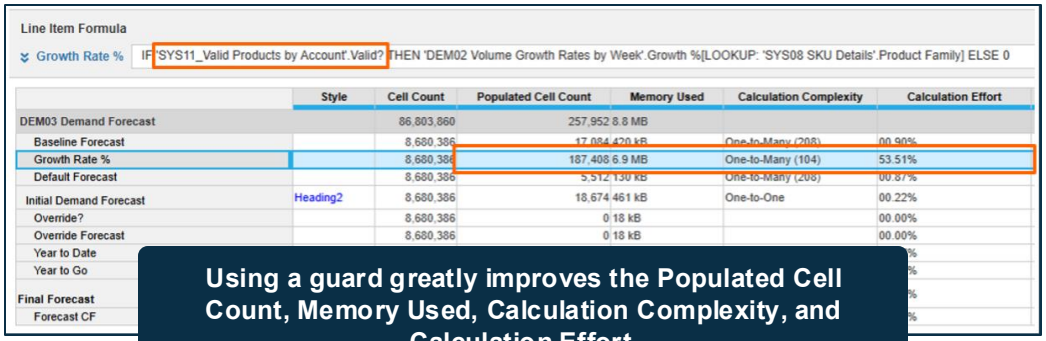
Polaris is already designed to minimize unnecessary calculations by skipping the default values of zero, blank and false, but it can only act on what it inherently knows.

When using guards to drive calculation efficiency, it’s not as simple as just adding an IF THEN statement to formulas. Thoughtful consideration about how and when to use guards is required.

Validation Modules: A tool for effective guards

A validation module is a small, targeted module that indicates intersections, such as which combinations of products and accounts, where a calculation should be performed.





Using a guard greatly improves the Populated Cell Count, Memory Used, Calculation Complexity, and Calculation Effort.

VALIDATION MODULE

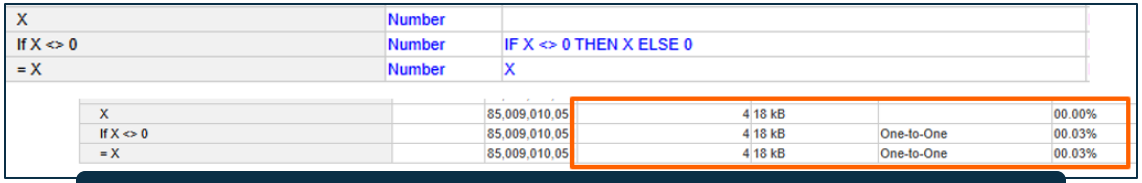
Contains the Boolean formatted line item 'Valid?' to indicate which combinations of products and accounts warrant a calculation, in this example the Growth Rate %, should be performed

INCORPORATING THE VALIDATION MODULE INTO A GUARD

The Growth Rate % formula has been updated to include an IF THEN ELSE statement that references the validation module. The calculation will only be performed for Product/Account combination where the Boolean is TRUE.

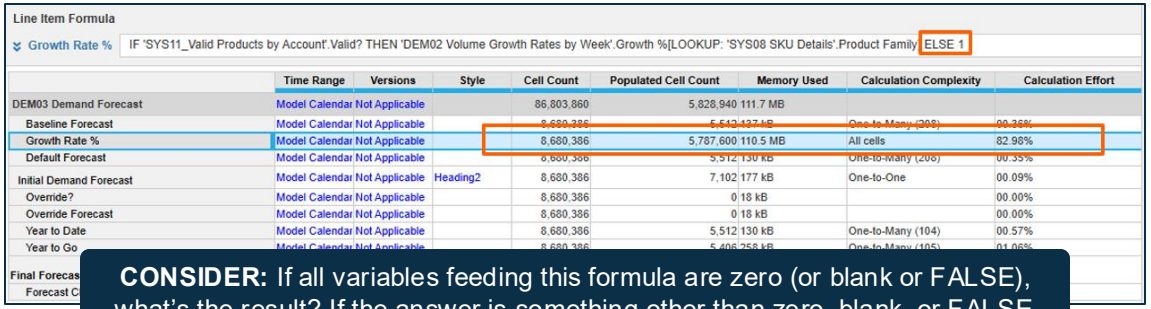
All guards are not created equal

Thoughtful consideration should be used when deciding when to use a guard. Guards are not always recommended or required, and they can inadvertently “reverse” the sparsity of a line item.



X	Number						
If X <> 0	Number	IF X <> 0 THEN X ELSE 0					
= X	Number	X					
X		85,009,010,05	4	18 kB			00.00%
If X <> 0		85,009,010,05	4	18 kB	One-to-One		00.03%
= X		85,009,010,05	4	18 kB	One-to-One		00.03%

NOTICE: The results are identical between these line items, meaning the IF/THEN portion is not needed.



Line Item Formula

Growth Rate % IF 'SYS11_Valid Products by Account'.Valid? THEN 'DEM02 Volume Growth Rates by Week'.Growth %[LOOKUP: 'SYS08 SKU Details'.Product Family] ELSE 1

	Time Range	Versions	Style	Cell Count	Populated Cell Count	Memory Used	Calculation Complexity	Calculation Effort
DEM03 Demand Forecast	Model Calendar	Not Applicable		86,803,860	5,828,940	111.7 MB		
Baseline Forecast	Model Calendar	Not Applicable		8,680,386	5,512	137 kB	One-to-Many (208)	00.36%
Growth Rate %	Model Calendar	Not Applicable		8,680,386	5,787,600	110.5 MB	All cells	82.98%
Default Forecast	Model Calendar	Not Applicable		8,680,386	5,512	130 kB	One-to-Many (208)	00.35%
Initial Demand Forecast	Model Calendar	Not Applicable	Heading2	8,680,386	7,102	177 kB	One-to-One	00.09%
Override?	Model Calendar	Not Applicable		8,680,386	0	18 kB		00.00%
Override Forecast	Model Calendar	Not Applicable		8,680,386	0	18 kB		00.00%
Year to Date	Model Calendar	Not Applicable		8,680,386	5,512	130 kB	One-to-Many (104)	00.57%
Year to Go	Model Calendar	Not Applicable		8,680,386	5,406,258	kB	One-to-Many (105)	01.06%
Final Forecast								
Forecast C								

CONSIDER: If all variables feeding this formula are zero (or blank or FALSE), what’s the result? If the answer is something other than zero, blank, or FALSE, then you should rethink your logic.

UNNECESSARY GUARDS

Some guards are redundant and add no performance benefit. In the Classic Anaplan engine, model builders often check for zeroes (a “<> 0” guard) to enable an early exit in IF statements. However, this isn’t required in Polaris since the engine inherently ignores zero values.

REVERSING SPARSITY

Formulas that result in a value for many or most cells can result in high density and poor performance at high dimensionality. Examples include formulas with guards ending in “… ELSE X + 1” or “… ELSE TRUE”.

For more detail, read this 5-minute article for information on **Applying Formula Guards to Optimize Calculations**.

Inline Conditions

What is an “inline condition”?

An “inline condition” refers to a formula condition that is written in the formula itself to reduce calculation effort.

SPLITTING SUB-EXPRESSIONS VS. USING INLINE CONDITIONS

Splitting sub-expressions into separate line items allows them to be shared, which can be beneficial for performance and maintenance. However, this forces the engine to perform a full calculation that may not be needed.

Since Polaris evaluates the entire formula to determine the most efficient calculation path, inline conditions can be used to reduce calculation effort. This becomes increasingly important as model dimensionality grows.

The key to understanding inline conditions is understanding a “cross” or “cross product”, which enumerates all the possible interactions of 2 or more things that aren’t actually related.

	Mon	Tues	Wed	Thu	Fri	Sat	Sun
Is Weekday	T	T	T	T	T	F	F

	UK	France	Spain	USA	Canada	Mexico
Speaks English	T	F	F	T	T	F

- **Line Item: Is Weekday = TRUE for days of the week**
- **Line Item: Speaks English = TRUE for countries whose primary language is English**

Is Included = Is Weekday AND Speaks English

	Mon	Tues	Wed	Thu	Fri	Sat	Sun
UK	T	T	T	T	T	F	F
France	F	F	F	F	F	F	F
Spain	F	F	F	F	F	F	F
USA	T	T	T	T	T	F	F
Canada	T	T	T	T	T	F	F
Mexico	F	F	F	F	F	F	F

Another line item, Is Included, is added. This line item is a “cross”.

Using inline conditions vs. splitting out sub-expressions

Because **Is Included** is a separate line item, the engine is forced to calculate every cell in that line item. However, if we are using **Is Included** in another line item called **Target,** we can put the conditions from **Is Included** directly inline in the formula for **Target** to take advantage of Polaris ignoring defaults.

Target = IF Is Weekday AND Speaks English THEN Units Sold ELSE 0

Target = IF Is Weekday AND Speaks English THEN Units Sold ELSE 0

	Mon	Tues	Wed	Thu	Fri	Sat	Sun
UK				3			
France							
Spain							
USA	8						
Canada							
Mexico							

Units Sold (Zeros shown as blanks)

Units Sold

	Mon	Tues	Wed	Thu	Fri	Sat	Sun
UK				3			
France		5					
Spain			1				
USA	8						
Canada							6
Mexico							

The engine will only calculate Target for intersections where Units Sold, Is Weekday and Speaks English are all non-default, which is a much more efficient approach than calculating Is Included independently for all cells. However, this requires that the conditions for Is Included are used inline when calculating Target.

Using inline conditions vs. splitting out sub-expressions (Continued)

The engine can only take the more efficient approach if the formula for **Is Included** is used inline when calculating **Target**. If it's instead its own line item, then the engine must calculate every combination.

Is Included split out (less performant):

Is Included = Is Weekday AND Speaks English
Target = IF Is Included THEN Units Sold ELSE 0

Is Included used inline (more performant):

Target = IF Is Weekday AND Speaks English THEN Units Sold ELSE 0

SO WHAT?

This example illustrates that, when deciding whether to use inline conditions or split sub-expressions into separate line items, you should look at **Calculation Complexity**. If you attempt to split out sub-expressions and…

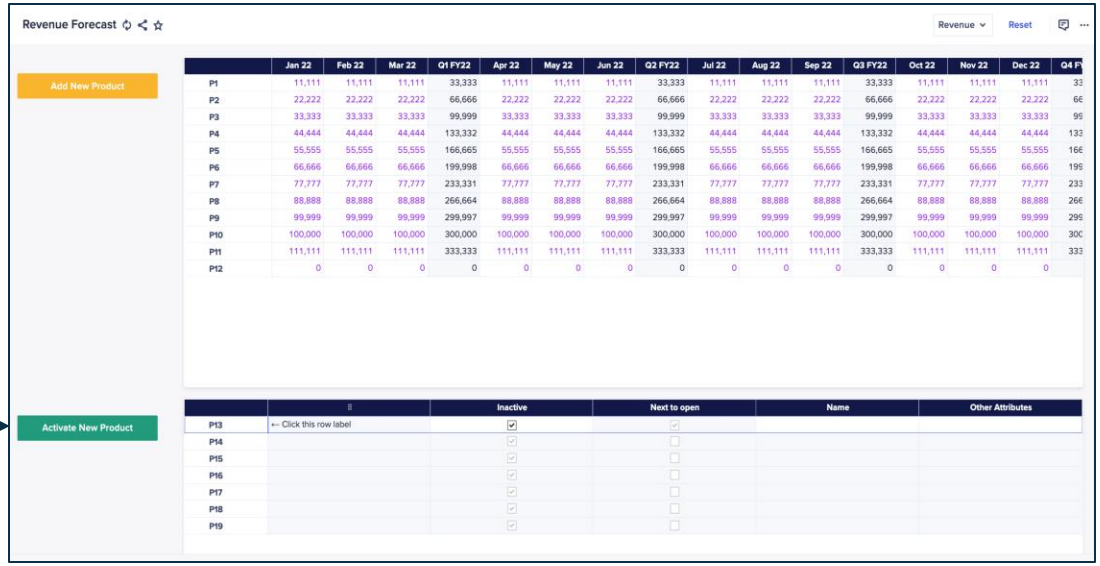
- **Calculation Complexity of the new line item is much higher,** then you should use inline conditions,
- **But if the calculation complexity is the same or less,** then it's ok to split out into separate line items.

For more detail, read this 5-minute article for information on **Optimizing Performance Using Inline Conditions**.

Inserting List Members

Manage Impact to Performance When Adding New Dimension Members

Adding new dimension members to a list will cause a full recalculation of any module containing that list, which at high dimensionality in Polaris can be substantial. One approach to reducing impact on performance is to activate pre-existing dimension members using in cell data instead.



Data write action to activate new items

Existing Products

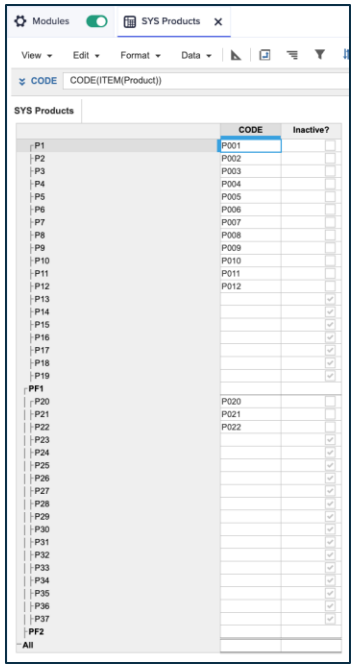
New Products

TIP 1: REVEAL NEW DIMENSION MEMBERS INDIVIDUALLY

Changing the value of a cell only recalculates formulas that refer to that cell. Use a Boolean as a filter to activate new members so the only recalculation happening is for the filter.

Manage Impact to Performance When Adding New Dimension Members

It's important to include empty, inactive list members within parent levels by default as changing a list item's parent will also cause a full module recalculation.



TIP 2: ADD EMPTY, INACTIVE LIST MEMBERS IN ADVANCE

Anticipate where new dimension members may be needed by including empty, inactive list members in each parent level by default. These items won't show up in reports or impact calculations, but they will be readily available for use without causing a full recalculation of modules that contain that list.

REMINDER: AVOID PEAK HOURS

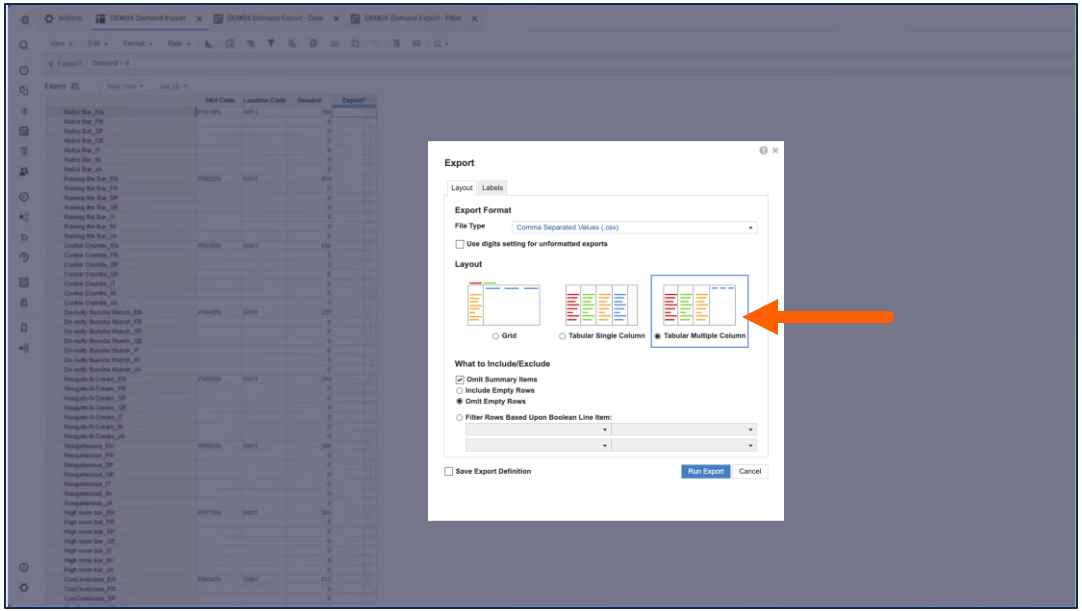
Major metadata additions should be performed in bulk during off hours to keep the model performance optimal during day-to-day use.

Watch this 5-minute video for an example of Inserting List Members.

Tabular Multiple Column Export

What is Tabular Multiple Column Export?

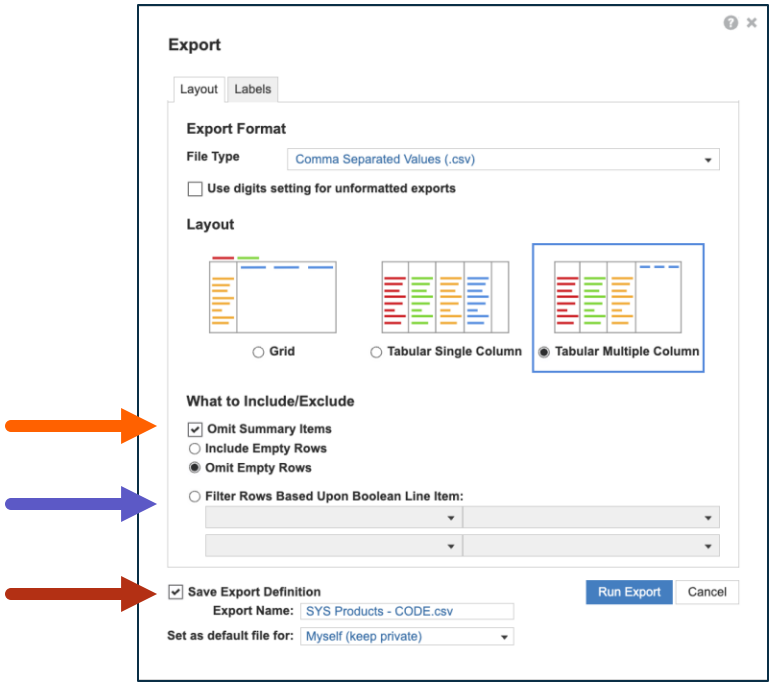
A more performant layout option for large exports that offers flexibility for future export maintenance, available in both Polaris and Classic



Leveraging the Tabular Multiple Column Export becomes even more important with Polaris, where datasets can be exponentially larger.

Available settings when using Tabular Multiple Column Export

After switching the layout to Tabular Multiple Column, additional options will become available that can be applied to the export and saved within the export action



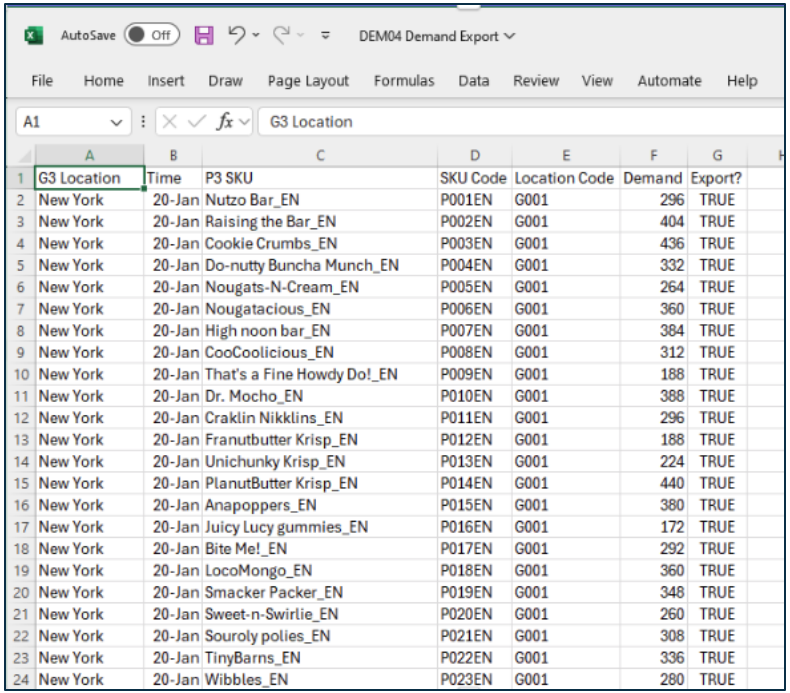
Omit Summary Items rather than using show/hide in the underlying module view for summary levels

Apply a filter within the action instead of within the module view

Save Export Definition saves all settings in the export action instead of within the module view, allowing easy updates in the future from the Actions pane

Considerations when using a Tabular Multiple Column Export

When using the Tabular Multiple Column layout, it's important to consider what's included in the file and the file size, and if needed, leverage options to achieve the desired output



	A	B	C	D	E	F	G
1	G3 Location	Time	P3 SKU	SKU Code	Location Code	Demand	Export?
2	New York	20-Jan	Nutzo Bar_EN	P001EN	G001	296	TRUE
3	New York	20-Jan	Raising the Bar_EN	P002EN	G001	404	TRUE
4	New York	20-Jan	Cookie Crumbs_EN	P003EN	G001	436	TRUE
5	New York	20-Jan	Do-nutty Buncha Munch_EN	P004EN	G001	332	TRUE
6	New York	20-Jan	Nougats-N-Cream_EN	P005EN	G001	264	TRUE
7	New York	20-Jan	Nougatacious_EN	P006EN	G001	360	TRUE
8	New York	20-Jan	High noon bar_EN	P007EN	G001	384	TRUE
9	New York	20-Jan	CooCoolicious_EN	P008EN	G001	312	TRUE
10	New York	20-Jan	That's a Fine Howdy Do!_EN	P009EN	G001	188	TRUE
11	New York	20-Jan	Dr. Mocho_EN	P010EN	G001	388	TRUE
12	New York	20-Jan	Craklin Nikklins_EN	P011EN	G001	296	TRUE
13	New York	20-Jan	Franutbutter Krisp_EN	P012EN	G001	188	TRUE
14	New York	20-Jan	Unichunky Krisp_EN	P013EN	G001	224	TRUE
15	New York	20-Jan	PlanutButter Krisp_EN	P014EN	G001	440	TRUE
16	New York	20-Jan	Anapoppers_EN	P015EN	G001	380	TRUE
17	New York	20-Jan	Juicy Lucy gummies_EN	P016EN	G001	172	TRUE
18	New York	20-Jan	Bite Me!_EN	P017EN	G001	292	TRUE
19	New York	20-Jan	LocoMongo_EN	P018EN	G001	360	TRUE
20	New York	20-Jan	Smacker Packer_EN	P019EN	G001	348	TRUE
21	New York	20-Jan	Sweet-n-Swirlie_EN	P020EN	G001	260	TRUE
22	New York	20-Jan	Souroly polies_EN	P021EN	G001	308	TRUE
23	New York	20-Jan	TinyBarns_EN	P022EN	G001	336	TRUE
24	New York	20-Jan	Wibbles_EN	P023EN	G001	280	TRUE

Only the rows where the export Boolean is true are included – the action is more flexible, with additional filters applied in the future as needed.

Tabular Multiple Column will export all line items within the module. If this isn't the desired behavior, create a separate model with only the necessary line items, as well as a separate module housing the export Boolean.

If the file size is large, use a separate filter module to keep the file size down.

Watch this 3-minute video for an example of Tabular Multiple Column Export.

Be the Next.